80

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/929,147	08/14/2001	Andreas H. Kuehnel	2007.015900	8938

22879 7590 08/10/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
TRUONG, CAM Y T	
ART UNIT	**PAPER NUMBER**
2162	

DATE MAILED: 08/10/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/929,147	Applicant(s) KUEHNEL, ANDREAS H.
	Examiner Cam Y T. Truong	Art Unit 2162

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 23 May 2005.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-17,19-24,26-31,33-38,40-52 and 54-57 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-17,19-24,26-31,33-38,40-52 and 54-57 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant filled Appeal Brief on 5/23/2005. Applicant's argument has been carefully considered by an appeal conference. Thus, the finality of the office action 6/2/2004 is withdrawn.

Claims 1-17, 19-24, 26-31, 33-38, 40-52, 54-57 are pending in this Office Action.

Response to Arguments

2. Applicant's arguments with respect to claims 1-17, 19-24, 26-31, 33-38, 40-52, 54-57 have been considered but are moot in view of the new ground(s) of rejection.

In view of the Appeal Brief filed on 5/23/2005, PROSECUTION IS HEREBY REOPENED. The rejections are set forth below.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) request reinstatement of the appeal.

If reinstatement of the appeal is requested, such request must be accompanied by a supplemental appeal brief, but no new amendments, affidavits (37 CFR 1.130, 1.131 or 1.132) or other evidence are permitted. See 37 CFR 1.193(b)(2).

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claim 43 recites the limitation "each word" in page 31, lines 3. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1-17, 19-21, 36-38, 40-42 and 57 are rejected under 35 U.S.C.101 because the language of the claim raises a question as to whether the claim is directed merely to an abstract idea that is not tied to a technological art, environment or machine which would result in a practice application producing a concrete, useful, and tangible result to form the basis of statutory subject matter under 35 U.S.C 101.

As regarding claim 1:

While the preamble of the claim states, "an apparatus for use in managing objects", the claim fails to contain a computer that is used implemented an apparatus for use in managing objects so as to realize its functionality. Thus, claim 1 is merely abstract idea whereby "a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" is being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claims 2-9 and 57 claim "the apparatus" and do not positively recite the method that is implemented by a machine. Thus, claims 2-9 are merely abstract idea and are being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claim 10:

While the preamble of the claim states, "an apparatus", the claim fails to contain a computer that is used implemented the apparatus so as to realize its functionality. Thus, claim 10 is merely abstract idea whereby "a usage counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" is being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claims 11-14 claim "the apparatus" and do not positively recite the method that is implemented by a machine. Thus, claims 11-14 are merely abstract idea and are being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claim 15:

While the preamble of the claim states, "a method for managing a plurality of clustered objects", the claim fails to contain a computer that is used implemented the method for managing a plurality of clustered objects so as to realize its functionality.

Thus, claim 15 is merely abstract idea whereby "a usage counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" is being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claims 16-17, 19-21 claim "the method" and do not positively recite the method that is implemented by a machine. Thus, claims 16-17 and 19-21 are merely abstract idea and are being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claim 36

While the preamble of the claim states, "a method for managing a plurality of clustered slots in a file", claim fails to contain a computer that is used implemented the method for managing a plurality of clustered slots in a file so as to realize its functionality. Thus, claim 36 is merely abstract idea whereby "a usage counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" is being processed without any links to a practical result in the technology arts and without computer manipulation.

As regarding claims 37-38, 40-42 claim "the method" and do not positively recite the method that is implemented by a machine. Thus, claims 37-38, 40-42 are merely abstract idea and are being processed without any links to a practical result in the technology arts and without computer manipulation.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 43, 44, 45 and 48 are rejected under 35 U.S.C. 102(b) as being anticipated by Macon, Jr. et al (or hereinafter "Macon") (US 5715455).

As to claim 43, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of

pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS.

In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

"consulting at least one usage counter that indicates how many sets of adjacent clusters are available storing data" as a counter count a number of contiguous clusters within a single group of clusters and it indicates which allocation units are free for assignment to a file (col. 6, lines 8-12);

"wherein each word comprises a plurality of bits" as clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains at least a word that comprises a plurality of bits.

As to claim 44, Macon teaches the claimed limitation:

"constructing the allocation data structure" as FAT file system (fig. 6);

"constructing the directory data structure" as directory (col. 4, lines 55-56).

As to claim 45, Macon teaches the claimed limitation "tracking in a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current

Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claim 48, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the slots" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635.

At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1, 3, 5, 7, 10, 22-24, 27, 36-38, 41, 50-52, 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lehman (US 6658437).

As to claim 1, Macon teaches the claimed limitations:

"a plurality of clusters " as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47), "each cluster comprising a plurality of objects" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"and a second data structure indicating the state of the clusters" as each cluster has a corresponding entry in the FAT that describes its current use: available, reserved, assigned to a file or unusable. For example, 0x0000 signifies an available cluster and 0xFFFF signifies an end-of cluster chain. FAT is represented as a second data structure (col. 4, lines 39-42).

"a first data structure indicating a state of the objects" as the root directory is known as the files area, which may be viewed as pools of clusters. Each cluster contains one or more sectors. Boot sector indicates reserved sectors, starting at 0 (two bytes) is represented as the state of sector (fig. 2, col. 4, lines 34-40).

"a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18).

Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "wherein a second data structure bit that is set indicates that one or more clusters of objects associated with said bit are free for storage of data". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 3, Macon teaches the claimed limitation "a plurality of containers populated by the cluster and wherein at least some containers comprises files" as the root directory contains files (col. 3, lines 30-35).

As to claim 5, Macon teaches the claimed limitation "the objects comprise slots in the file" as each sector having a plurality of storage locations (col. 3, lines 18-19).

As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4).

As to claim 10, Macon teaches the claimed limitations:

"a plurality of files" as the root directory is the root of all files /subdirectories (col. 4, line 55-56);

"a plurality of clusters populating each file," as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47) "each cluster comprising a plurality of slots" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"a directory bitmap indicating the state of the clusters" as in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined

by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at OxFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (OxFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47);

as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains

8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprise a plurality of bits.

Macon does not explicitly teach the claimed limitation "a bit in the directory bimap being set to indicate whether a cluster associated with said bit is free; an allocation bitmap indicating a state of the slots". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 22, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit;

Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer

to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes

(col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "to indicate that associated objects are free for storing data". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 23 , Macon teaches the claimed limitations:

"constructing the first data structure" as FAT file system (fig. 6);

"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claim 24, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the states of clusters of objects in the second data structure includes tracking a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the

position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster

(0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claim 27, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a

sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 36, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken

to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address

at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits.

Macon does not explicitly teach the claimed limitation "thereby indicating which clusters of objects are free for storing data". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claims 37 and 51, Macon teaches the claimed limitation:

"constructing the allocation data structure" as FAT file system (fig. 6);

"constructing the directory data structure" as directory (col. 4, lines 55-56).

As to claims 38 and 52, Macon teaches the claimed limitation "tracking in a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current

Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 41, 48 and 55, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the slots" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision

block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 50, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value;

otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the first and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to

serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the directory data structure" a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains

8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "thereby indicating which clusters are free for storing data". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

11. Claims 2, 4 and 11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lehman (US 6658437)and further in view of Lehman (USP 5732402).

As to claim 2, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "a plurality of containers populated by the clusters and control data associated with the containers". However, Lehman teaches that management of the LOB data space, including allocation of space and storage/retrieval

of data, is controlled by allocation pages. Allocation pages are represented as a plurality of container (col. 5, lines 43-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman's teaching of management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by allocation pages to Macon in order to set flags in storage for indicating whether a space is currently occupied or is free to be used.

As to claim 4 Macon discloses the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon's system in order to store a large amount objects in the context of a paging memory.

As to claim 11, Macon discloses the claimed limitation subject matter in claim 10, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon in order to store a large amount objects in the context of a paging memory.

12. Claims 6 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lehman (US 6658437) and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 6, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 objects". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon in order to store a large data object in a cluster.

As to claim 12, Macon discloses the claimed limitation subject matter in claim 10, except the claimed limitation "wherein each cluster comprises 16 slots". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon's system in order to store a large data object in a cluster.

13. Claims 8, 13, 26, 33, 40, and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Lehman (US 6658437) and further in view of Zwilling et al (or hereinafter "Zwilling").

As to claims 8, 26, 33, 40, and 54, Macon discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used

and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

As to claim 13, Macon, Shoroff disclose the claimed limitation subject matter in claim 10, except the claimed limitation "at least one other counter selected from the group consisting of : a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and to Macon, Shoroff in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

14. Claims 15-17, 20, 29, 30, 31, 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Lehman (US 6658437) and Grossier (US 6553478).

As to claim 15, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit;

Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer

to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes

(col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

Macon does not explicitly teach the claimed limitation " each bit in the second data structure being set to indicate whether a cluster of objects associated with said bit is free; "wherein each word comprises a plurality of bits associated with an implementation specific wordlength". Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45). Grossier teaches data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word (col. 3, lines 45-52).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy and Grossier's teaching of data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple

number of bytes for each word to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claims 16 and 30, Macon teaches the claimed limitations:
"constructing the first data structure" as FAT file system (fig. 6);
"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claims 17 and 31, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the states of clusters of objects in the second data structure includes tracking a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to

MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set.. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claim 20, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad

cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 29, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to

640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as

constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits.. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

Macon does not explicitly teach the claimed limitation "a set bit in the second data structure indicating that an associated cluster of objects is free for storing data; wherein the words each have a wordlength based on a maximum number of bits handled by a processor that executes an operating system".

Lehaman teaches each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy (col. 10, lines 40-45). Grossier teaches data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word (col. 3, lines 45-52).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehaman's teaching of each bit directly shows the status of each individual block. Adjacent 0's show larger sections of free blocks. Adjacent 1's shows blocks that are busy and Grossier's teaching of data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 34, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event

that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

15. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Lehman (US 6658437) and Grossier and further in view of Zwilling et al (or hereinafter "Zwilling") (US 6249792).

As to claim 19, Macon discloses the claimed limitation subject matter in claims 1, 15, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file (col. 11, lines 65-67; col. 12, lines 1-5).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

16. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view Grossier and Lehman (US 6658437) and further in view of Zwilling et al (or hereinafter "Zwilling").

As to claim 33, Macon discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

17. Claim 47 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Zwilling.

As to claim 47, Macon discloses the claimed limitation subject matter in claim 43, except the claimed limitation "at least one other counter selected from the group

consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

18. Claim 57 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view Lehman (US 6658437) and further in view of Bilbrey et al (or hereinafter "bilbrey") (USP 5227863).

As to claim 57, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "wherein the number of sets of adjacent bits is selected from the group consisting of 2, 4, 8, 16, 32 and 64". Bilbrey teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits (col. 40, lines 60-67).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Bilbrey's teaching of teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits to Macon in order to store or display data on a pixel by pixel basis.

19. Claims 1, 3, 5, 7, 10, 22-24, 27, 36-38, 41, 50-52, 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Shoroff et al (or hereinafter "Shoroff") (US 6023744).

As to claim 1, Macon teaches the claimed limitations:

"a plurality of clusters " as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47), "each cluster comprising a plurality of objects" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"and a second data structure indicating the state of the clusters" as each cluster has a corresponding entry in the FAT that describes its current use: available, reserved, assigned to a file or unusable. For example, 0x0000 signifies an available cluster and 0xFFFF signifies an end-of cluster chain. FAT is represented as a second data structure (col. 4, lines 39-42).

"a first data structure indicating a state of the objects" as the root directory is known as the files area, which may be viewed as pools of clusters. Each cluster contains one or more sectors. Boot sector indicates reserved sectors, starting at 0 (two bytes) is represented as the state of sector (fig. 2, col. 4, lines 34-40).

"a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data,

i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "wherein a second data structure bit that is set indicates that one or more clusters of objects associated with said bit are free for storage of data". Shoroff's teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space to Macon's system in order to track or allocate data in memory correctly and reduce

number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 3, Macon teaches the claimed limitation "a plurality of containers populated by the cluster and wherein at least some containers comprises files" as the root directory contains files (col. 3, lines 30-35).

As to claim 5, Macon teaches the claimed limitation "the objects comprise slots in the file" as each sector having a plurality of storage locations (col. 3, lines 18-19).

As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4).

As to claim 10, Macon teaches the claimed limitations:

"a plurality of files" as the root directory is the root of all files /subdirectories (col. 4, line 55-56);

"a plurality of clusters populating each file," as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47) "each cluster comprising a plurality of slots" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"a directory bitmap indicating the state of the clusters" as in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47);

as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprise a plurality of bits.

Macon does not explicitly teach the claimed limitation "a bit in the directory bimap being set to indicate whether a cluster associated with said bit is free; an allocation bitmap indicating a state of the slots". Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a

bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 22, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad

cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision

block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18).

Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "to indicate that associated objects are free for storing data". Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 23, Macon teaches the claimed limitations:

"constructing the first data structure" as FAT file system (fig. 6);

"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claim 24, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the states of clusters of objects in the second data structure includes tracking a bitmap" as a request

to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster

(0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claim 27, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 36, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number

of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for

clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits.

Macon does not explicitly teach the claimed limitation "thereby indicating which clusters of objects are free for storing data". Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claims 37 and 51, Macon teaches the claimed limitation:

"constructing the allocation data structure" as FAT file system (fig. 6);

"constructing the directory data structure" as directory (col. 4, lines 55-56).

As to claims 38 and 52, Macon teaches the claimed limitation "tracking in a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address

at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 41, 48 and 55, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the slots" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 50, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the first and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as

constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the directory data structure" a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

"wherein each word comprises a plurality of bits" as Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). Since a cluster contains 2048 bytes that contain more than 32 bits; thus, the cluster can contains many words that comprises a plurality of bits.

Macon does not explicitly teach the claimed limitation "thereby indicating which clusters are free for storing data". Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

20. Claims 2, 4 and 11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Shoroff and further in view of Lehman (USP 5732402).

As to claim 2, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "a plurality of containers populated by the clusters and control data associated with the containers". However, Lehman teaches that management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by allocation pages. Allocation pages are represented as a plurality of container (col. 5, lines 43-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman's teaching of management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by allocation pages to Macon in order to set flags in storage for indicating whether a space is currently occupied or is free to be used.

As to claim 4 Macon discloses the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon's system in order to store a large amount objects in the context of a paging memory.

As to claim 11, Macon discloses the claimed limitation subject matter in claim 10, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating

free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon in order to store a large amount objects in the context of a paging memory.

21. Claims 6 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Shoroff and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 6, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 objects". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon in order to store a large data object in a cluster.

As to claim 12, Macon discloses the claimed limitation subject matter in claim 10, except the claimed limitation "wherein each cluster comprises 16 slots". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon's system in order to store a large data object in a cluster.

22. Claims 8, 13, 26, 33, 40, and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Shoroff and further in view of Zwilling et al (or hereinafter "Zwilling").

As to claims 8, 26, 33, 40, and 54, Macon discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

As to claim 13, Macon, Shoroff disclose the claimed limitation subject matter in claim 10, except the claimed limitation "at least one other counter selected from the

group consisting of : a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and to Macon, Shoroff in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

23. Claims 15-17, 20, 29, 30, 31, 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Shoroff and Grossier (US 6553478).

As to claim 15, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken

where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of

the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

Macon does not explicitly teach the claimed limitation " each bit in the second data structure being set to indicate whether a cluster of objects associated with said bit is free; "wherein each word comprises a plurality of bits associated with an implementation specific wordlength". Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10). Grossier teaches data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word (col. 3, lines 45-52).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space and Grossier's teaching of data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claims 16 and 30, Macon teaches the claimed limitations:

"constructing the first data structure" as FAT file system (fig. 6);

"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claims 17 and 31, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the states of clusters of objects in the second data structure includes tracking a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of

the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster
(0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claim 20, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a

sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 29, Macon teaches the claimed limitations:

"tracking states of clusters of objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the

number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at OxFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system

tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67);

"consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the second data structure" as a counter count a number of contiguous clusters in FAT structure. Clusters are linked together. A cluster size was 2048 bytes (col. 6, lines 5-12; col. 4, lines 63-67). One byte contains 8 bits. A word is the largest amount of data, i.e., word sizes of 16 bits and 32 bits (Computer Dictionary, page 52, col. Left, lines 11-14; page 510, col. Right, lines 14-18). As seen above, a cluster contains bits that are set in words of FAT. The number of contiguous clusters are presented as a number sets of adjacent bits;

Macon does not explicitly teach the claimed limitation "a set bit in the second data structure indicating that an associated cluster of objects is free for storing data; wherein the words each have a wordlength based on a maximum number of bits handled by a processor that executes an operating system".

Shoroff teaches the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space (fig. 2, col. 5, lines 7-10).

Grossier teaches data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word (col. 3, lines 45-52).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Shoroff's teaching of the master file table contains a bitmap record 52 having a bit for each cluster on the volume, with each bit having a value representative of whether a cluster is allocated to a file or is free space and Grossier's teaching of data is stored in the memory 43 and 45 in bit sequences forming words, each word consisting of a multiple number of 8 bit sequences or bytes. In this particular example the wordlength used is 32 bits but other examples may consist of 16 bits or 64 bits or any other multiple number of bytes for each word to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data and protect against system failures for adding the physical locations to free space.

As to claim 34, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the

unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

24. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Shoroff and Grossier and further in view of Zwilling et al (or hereinafter "Zwilling") (US 6249792).

As to claim 19, Macon discloses the claimed limitation subject matter in claims 1, 15, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file (col. 11, lines 65-67; col. 12, lines 1-5).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used

and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

25. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view Grossier and Shoroff and further in view of Zwilling et al (or hereinafter "Zwilling").

As to claim 33, Macon discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

26. Claim 47 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Zwilling.

As to claim 47, Macon discloses the claimed limitation subject matter in claim 43, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how

many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file.

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file to Macon in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

27. Claim 57 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view shoroff and further in view of Bilbrey et al (or hereinafter "bilbrey") (USP 5227863).

As to claim 57, Macon discloses the claimed limitation subject matter in claim 1, except the claimed limitation "wherein the number of sets of adjacent bits is selected from the group consisting of 2, 4, 8, 16, 32 and 64". Bilbrey teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits (col. 40, lines 60-67).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Bilbrey's teaching of teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits to Macon in order to store or display data on a pixel by pixel basis.

Allowable Subject Matter

28. Claims 9, 14, 21, 28, 35, 42, 49 and 56 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

As to claims 9 and 14, none of the available prior art of record teaches or fairly suggest "the second data structure contains clusters of at least four adjacent free bits; the second data structure is not empty, but contains no clusters of four adjacent free bits; the second data structure is empty, but allocation bitmap still shows free pages".

As to claims 21, 28, 35, 42, 49 and 56, none of the available prior art of record teaches or fairly suggest "a first list containing information indicating that the directory data structure for files in this list contains clusters of at least four adjacent free bits; a second list containing information indicating that the directory data structure for files in this list is not empty, but contains no clusters of four adjacent free bits; a third list containing information indicating that the directory data structure for files in this list is empty, but allocation bitmap still shows free slots".

Conclusion

29. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure

Holzhammer et al (US 5630093).

Contact Information

30. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cam Y T Truong whose telephone number is. (571) 272-4042. The examiner can normally be reached on Monday to Firday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene can be reached on (571) 272-4107. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Cam-Y Truong
Patent Examiner
Art Unit 2162
1/19/2005

JEAN M. CORRIELUS
PRIMARY EXAMINER